CLAIMS

1. A method comprising:

receiving an indication of an application programming interface definition that is written in a markup language; and

transforming the interface definition into a non-markup language source file.

2. A method as recited in claim 1, wherein the source file comprises a component object model (COM) header file.

3. A method as recited in claim 1, wherein the source file comprises code mapping a set of enumeration values to corresponding string values.

4. A method as recited in claim 1, further comprising transforming the application programming interface definition into a documentation file that describes the contents of the non-markup language source file.

5. A method as recited in claim 1, wherein the source file comprises a proxy object code file.

6. A method as recited in claim 1, further comprising transforming the application programming interface definition into a test proxy object code file, wherein the test proxy object code file includes a plurality of test proxies to assist in testing the non-markup language source file.

7. A method as recited in claim 1, wherein receiving the indication comprises receiving a filename of the application programming interface definition.

8. A method as recited in claim 1, wherein receiving the indication comprises receiving the application programming interface definition.

9. A method comprising:

receiving an indication of an interface definition, wherein the interface definition includes a plurality of constructs;

transforming the interface definition into data for a first file; and

transforming the interface definition into data for a second file, wherein the data for the first file is different than the data for the second file.

10. A method as recited in claim 9, wherein the first file and the second file are each different ones of the following types of files: a component object model (COM) header file, a mapping file to map enumeration values to corresponding string values, a proxy object code file, and a documentation file.

11. A method as recited in claim 9, further comprising:

determining that the interface definition has been changed;

re-transforming the interface definition into data for a third file, wherein the third file is the same type of file as the first file; and

re-transforming the interface definition into data for a fourth file, wherein the fourth file is the same type of file as the second file.

12. A method as recited in claim 11, wherein determining that the interface definition has been changed comprises automatically detecting that the interface definition has been changed.

13. One or more computer readable media having stored thereon a plurality of instructions that, when executed by a transformation engine, causes the transformation engine to:

access a plurality of constructs in an application programming interface description, wherein the description is written in an extensible markup language (XML) format; and

transform each of the plurality of constructs into code for a component object module (COM) application programming interface header file.

14. One or more computer readable media as recited in claim 13, wherein the transformation engine comprises a series of instructions executed by one or more processors.

15. One or more computer readable media as recited in claim 13, wherein the plurality of instructions include instructions to:

check whether a declare enumeration construct is to be transformed into a series of manifest constants or into a component object model enumeration declaration; and

transform the enumeration construct into either the series of manifest constants or the component object model enumeration declaration based on the checking.

16. One or more computer readable media as recited in claim 13, wherein the plurality of instructions include instructions to transform a declare enumeration construct into a series of manifest constants.

17. One or more computer readable media as recited in claim 13, wherein the plurality of instructions include instructions to transform a declare enumeration construct into a component object model enumeration declaration.

18. One or more computer readable media as recited in claim 13, wherein the plurality of instructions include instructions to transform a declare function construct into a component object model function declaration.

19. One or more computer readable media as recited in claim 13, wherein the plurality of instructions include instructions to transform a declare class object construct into a component object model class object ID declaration.

20. One or more computer readable media as recited in claim 13, wherein the plurality of instructions include instructions to transform a declare interface construct into a component object model forward class declaration.

21. One or more computer readable media as recited in claim 13, wherein the plurality of instructions include instructions to transform a declare data structure construct into a component object model data structure declaration.

22. One or more computer readable media as recited in claim 13, wherein the plurality of instructions include instructions to transform a declare macro construct into a component object model manifest constant.

23. A method comprising:

receiving an indication of an application programming interface description, wherein the description is written in an extensible markup language (XML) format;

identifying a plurality of constructs in the application programming interface description; and

transforming each of the plurality of constructs into code for a component object module (COM) application programming interface header file.

24. A method as recited in claim 23, wherein the code for the component object module comprises C source code.

25. A method as recited in claim 23, wherein the code for the component object module comprises C++ source code.

26. A method as recited in claim 23, wherein the transforming comprises transforming a declare enumeration construct into a series of manifest constants.

27. A method as recited in claim 23, wherein the transforming comprises transforming a declare enumeration construct into a component object model enumeration declaration.

28. A method as recited in claim 23, wherein the transforming comprises transforming a declare function construct into a component object model function declaration.

29. A method as recited in claim 23, wherein the transforming comprises transforming a declare class object construct into a component object model class object ID declaration.

30. A method as recited in claim 23, wherein the transforming comprises transforming a declare interface construct into a component object model forward class declaration.

31. A method as recited in claim 23, wherein the transforming comprises transforming a declare data structure construct into a component object model data structure declaration.

32. A method as recited in claim 23, wherein the transforming comprises transforming a declare macro construct into a component object model manifest constant.

33. One or more computer readable media having stored thereon a plurality of instructions that, when applied by a transformation engine, causes the transformation engine to perform acts comprising:

receiving an indication of an application programming interface description, wherein the description is written in an extensible markup language (XML) format;

identifying one or more enumeration declaration constructs in the application programming interface description; and

transforming each of the enumeration declaration constructs into a mapping of enumeration values to corresponding string values.

34. One or more computer readable media as recited in claim 33, wherein the transformation engine comprises a series of instructions executed by one or more processors.

35. One or more computer readable media having stored thereon a plurality of instructions that, when applied by a transformation engine, causes the transformation engine to perform acts comprising:

receiving an indication of an application programming interface description, wherein the description is written in an extensible markup language (XML) format;

identifying a plurality of constructs in the application programming interface description; and

transforming each of the plurality of constructs into code for a test proxy file to be used in testing a component object module (COM) application programming interface header file generated from the application programming interface description.

36. One or more computer readable media as recited in claim 35, wherein the transformation engine comprises a series of instructions executed by one or more processors.

37. One or more computer readable media as recited in claim 35, wherein each of the plurality of constructs comprises a declare interface construct.

38. A computer-readable medium having stored thereon a data structure comprising:

an id attribute field that contains data identifying an application programming interface description construct;

a plurality of construct fields that contain data describing the application programming interface; and

a field functioning to identify the end of the data structure.

39. A computer-readable medium as recited in claim 38, wherein the plurality of construct fields include one or more declare enumeration construct fields.

40. A computer readable media as recited in claim 39, wherein each declare enumeration construct field includes:

a plurality of declare enumeration member constructs; and

an enumeration flag attribute that identifies whether the plurality of declare enumeration member constructs are to be transformed into a series of manifest constants or transformed into a component object model enumeration declaration.

41. A computer-readable medium as recited in claim 38, wherein the plurality of construct fields include one or more declare function construct fields.

42. A computer-readable medium as recited in claim 38, wherein the plurality of construct fields include one or more declare class object construct fields.

43. A computer-readable medium as recited in claim 38, wherein the plurality of construct fields include one or more declare interface construct fields.

44. A computer readable media as recited in claim 43, wherein each declare interface construct field includes:

one or more declare method constructs, wherein each declare method construct stores data identifying a method corresponding to the interface defined by the declare interface construct field; and

wherein each declare method construct includes one or more declare parameter construct fields, wherein each declare parameter construct field stores data identifying a parameter of the method.

45. A computer-readable medium as recited in claim 38, wherein the plurality of construct fields include one or more declare data structure construct fields.

46. A computer-readable medium as recited in claim 38, wherein the plurality of construct fields include one or more declare macro construct fields.